

Multivariate analysis for Maxima in High Dimensions

Jérémy Barbay and Javiel Rojas

Departamento de Ciencias de la Computación, Universidad de Chile, Chile
{jeremy, jrojas}@dcc.uchile.cl

Abstract We study the problem of computing the MAXIMA of a set of n d -dimensional points. For dimensions 2 and 3, there are algorithms to solve the problem with order-oblivious instance-optimal running time. However, in higher dimensions there is still room for improvements. We present an algorithm sensitive to the structural entropy of the input set, which improves the running time, for large classes of instances, on the best solution for MAXIMA to date for $d \geq 4$.

1 Introduction

The problem of computing the MAXIMA of a set \mathcal{S} of points was first formulated in 1974 by Kung et al. [9]: a point from \mathcal{S} is called *maximal* if none of the remaining vectors in \mathcal{S} dominates it in every component, and the MAXIMA of \mathcal{S} (denoted by $M(\mathcal{S})$) is the set of maximal points in \mathcal{S} . For any constant dimension d this problem can be solved naively in time within $\mathcal{O}(n^2)$ by comparing every possible pair of points. Kung et al. [9] proposed two different algorithms to solve the problem in two and three dimensions, respectively, running in time within $\mathcal{O}(n \log n)$. For higher dimensions, Kung et al. [9] presented a divide-and-conquer algorithm running in time within $\mathcal{O}(n \log^{d-2} n)$ for dimensions $d > 3$, and showed a lower bound of within $\Omega(n \log n)$ for this problem for any dimension $d > 2$.

In 1985, Kirkpatrick and Seidel [8] gave the first output-size sensitive algorithm for this problem, running in time within $\mathcal{O}(n \log h)$ in the plane, and within $\mathcal{O}(n \log^{d-2} h)$ for dimension $d > 3$, where h is the size of the MAXIMA. In 2 and 3 dimensions, this result remained unbeaten for almost 25 years until Afshani et al. [2] described an *instance-optimal* algorithm for this problem (i.e., an algorithm whose cost is at most a constant factor from the cost of any other algorithm running on the same input, for every input instance).

Given this improvement in 2 and 3 dimensions, one natural question is whether the same technique can be applied to higher dimensions in order to improve upon Kirkpatrick and Seidel’s results [8] in dimension $d \geq 4$. We show that the upper bound can indeed be applied to higher dimensions, even though the generalization of the order-oblivious instance-optimality result is still open due to the lack of advanced lower bound techniques in high dimensions. We introduce some basic definitions in Section 2; and describe our generalization of Afshani et al.’s upper bound [2] to high dimension in Section 3, before discussing the potential for instance-optimality results in high dimension in Section 4.

2 Preliminaries

An algorithm is said to be *instance-optimal* if its cost is at most a constant factor from the cost of any other algorithm on the same input, for every input instance. As such algorithm does not always exist, it is often useful to consider a relaxation of this concept, order-oblivious instance-optimality: an algorithm is said to be *order-oblivious instance-optimal* if it is instance-optimal on the worst input order possible.

Afshani et al. [2] improved on the analysis of Kirkpatrick and Seidel [8] for the computation of the MAXIMA in dimensions 2 and 3 refining it to order-oblivious instance-optimal algorithms. In 2 dimensions, they showed that a simple variant of the output sensitive algorithm originally described by Kirkpatrick and Seidel [8] is, surprisingly, order-oblivious instance-optimal. For the 3 dimensional case, they described a completely new algorithm and a proof of its order-oblivious instance-optimality.

To refine the analysis, Afshani et al. [2] introduced the notion of *structural entropy* of a set \mathcal{S} of points, a measure of difficulty of the input instances of MAXIMA, and described two algorithms sensitive to this measure. To define the structural entropy of a point-set, Afshani et al. [2] first introduced the concept of *respectful partition*:

Definition 1 (Respectful Partition). *A partition Π of a set \mathcal{S} of points in \mathbb{R}^d into disjoint subsets $\mathcal{S}_1, \dots, \mathcal{S}_t$ is said to be respectful if each subset \mathcal{S}_k is either a singleton or can be enclosed by a d -dimensional axis-aligned box B_k whose interior is completely below the MAXIMA of \mathcal{S} .*

Intuitively, the entropy of a respectful partition is the minimum number of bits required to encode it, and the structural entropy of a point-set is the entropy of a respectful partition with minimal entropy. Formally, the structural entropy of a point-set is defined as follows:

Definition 2 (Structural Entropy). *The entropy $\mathcal{H}(\Pi)$ of a partition Π of \mathcal{S} into disjoint subsets $\mathcal{S}_1, \dots, \mathcal{S}_t$ is defined as the value $\sum_{k=1}^t (|\mathcal{S}_k|/n) \log(n/|\mathcal{S}_k|)$. The structural entropy $\mathcal{H}(\mathcal{S})$ of the input set \mathcal{S} is the minimum of $\mathcal{H}(\Pi)$ over all respectful partitions Π of \mathcal{S} .*

Afshani et al. [2] showed that the MAXIMA of a point-set in dimensions 2 and 3, can be computed in time within $\mathcal{O}(n(\mathcal{H}(\mathcal{S}) + 1))$, and proved a matching lower bound for any algorithm which ignores the input order.

In Section 3 we generalize to higher dimensions Afshani et al. [2]’s upper bound on the computational complexity of MAXIMA in the worst case over instances of fixed size and structural entropy. This yields an algorithm sensitive to the structural entropy of the input set, hence improving on the solution described by Kirkpatrick and Seidel [8] for $d \geq 4$.

3 Multivariate analysis for the computation of Maxima

We present the algorithm **DPC-Maxima** (where DPC stands for “Divide, Prune, and Conquer”) which computes the MAXIMA of a d -dimensional set \mathcal{S} of points with running time sensitive to the structural entropy of the input set.

Similar to Afshani et al.’s algorithm [2], **DPC-Maxima** performs several pruning steps removing from \mathcal{S} points that are detected to be dominated, until the remaining set becomes the MAXIMA of the original set. For this, the algorithm repeatedly partitions \mathcal{S} into an increasing number of subsets, and for each partition removes from \mathcal{S} the subsets that are contained in a box completely below the MAXIMA of \mathcal{S} . The following is an outline of **DPC-Maxima**:

Algorithm 1 DPC-Maxima

Input: A set S of n points in \mathbb{R}^d

Output: The MAXIMA of \mathcal{S}

- 1: $j \leftarrow 0, \mathcal{M} \leftarrow \{S\}$
 - 2: **while** there is a set in \mathcal{M} with more than one point **do**
 - 3: $j \leftarrow j + 1, r_j \leftarrow 2^{2^j}, \mathcal{M} \leftarrow \{\}$
 - 4: partition S into r_j subsets S_1, S_2, \dots, S_{r_j} of size at most n/r_j
 - 5: **for** $i = [1..r_j]$ **do**
 - 6: Let I_i be the minimum axis-aligned enclosing rectangle of S_i
 - 7: **if** I_i is below the MAXIMA of \mathcal{S} **then** remove all points in S_i from S
 else add S_i to \mathcal{M}
 - 8: **return** the points remaining in S
-

Note that **DPC-Maxima** always terminates: due to the restriction on the size of the subsets (in step 4), at most $\log \log n$ iterations of the outer loop will be performed for any set S of n points. Besides, the algorithm is correct: only dominated points are removed, and in the last iteration of the outer loop every subset is a singleton, so the points remaining in \mathcal{S} after that iteration are precisely those in the MAXIMA of \mathcal{S} .

In the rest of the section we analyze the running time of **DPC-Maxima**. Two main issues need to be addressed: how partitions are chosen in step 4, and how the filter steps 5-7 are performed efficiently. Partitions need to be chosen with care: if B is a box in an optimal respectful partition of \mathcal{S} , the subsets considered by the algorithm that contain points within B need to be small (compact) enough so they fall completely inside B (and the subsets get pruned), but large enough that only a small number of subsets is needed to prune all the points within B . Besides, for a given partition, **DPC-Maxima** needs to check in small time whether the minimum axis-aligned enclosing rectangle of each subset in the partition is below the MAXIMA of \mathcal{S} .

We address first the selection of the partition. We show in the following lemma that any point set S can be partitioned into subsets so that the faces of any axis aligned box B intersects a small number of them. In particular, if B is a box in a respectful partition of S , the lemma provides a bound on the number of points in $S \cap B$ remaining after the filtering in steps 5-7 of the algorithm **DPC-Maxima**.

Lemma 1. *Let \mathcal{S} be a set of n points in \mathbb{R}^d , and r be an integer in $[1..n]$. There is a partition Π of \mathcal{S} into r subsets such that each subset has at most n/r points, and for any axis aligned d -dimensional box B :*

- i. B partially intersects (i.e. intersects but does not contain) at most $\mathcal{O}(r^{1-1/d})$ subsets in Π*
- ii. There are at most $\min\{|\mathcal{S} \cap B|, \mathcal{O}(n/r^{1/d})\}$ points within B that belong to subsets in Π not fully contained within B*

Besides, this partition can be found in time within $\mathcal{O}(|\mathcal{S}| \log r)$.

Proof. The partition Π can be obtained building a k - d tree [4], using the median to split at each level, and recursing until the number of points within each cell drops to n/r or less. Built like this, the k - d tree will have at most $\log r$ levels and r leaf cells, each containing at most n/r points. For each leaf cell in the k - d tree, the subset of points contained within it is added to Π . If needed, empty subsets are added also to obtain the total r subsets. This way of proceeding requires running time within $\mathcal{O}(|\mathcal{S}|)$ for each level, for a total running time within $\mathcal{O}(|\mathcal{S}| \log r)$.

To prove proposition (i), we use the fact that any axis aligned d -dimensional box intersects at most $\mathcal{O}(r^{1-1/d})$ leaf cells in a k - d tree with $\mathcal{O}(r)$ leaf cells [4]. Since the subsets in Π fall within cells in a k - d tree with at most r leaf cells, the result follows. Finally, (ii) derives from (i) and from the fact that every subset has at most n/r points: the points within B belong, in the partition, to subsets that are completely contained within B , or partially intersecting B ; the latest ones can be at most $\mathcal{O}(r^{1-1/d})$ because of (i), and contain at most n/r points each, for a total number of points within $\mathcal{O}(n/r^{1/d})$, which completes the proof. \square

Now, we turn the attention to the filtering process in steps 5-7 of the algorithm **DPC-Maxima**. For this, we introduce a data structure that answers dominance queries over a point set, and that benefits from knowing a priori the number of queries to perform. With it, the subsets in the partition that fall within boxes completely below the MAXIMA can be detected in small time.

Lemma 2. *Given a set S of n points in \mathbb{R}^d , and an integer parameter $r \in [1..n]$, there is a data structure that answers r dominance queries over S (i.e. given a query point q , detecting whether exists a point $p \in S$ that dominates q) in total time and space within $\mathcal{O}(n \log^{d-2} r)$.*

Proof. Afshani et al. [2] introduced a simple data structure that answers dominance queries over a point-set of size n in time within $\mathcal{O}(\log^{d-2} n)$, with preprocessing time within $\mathcal{O}(n \log^{d-2} n)$. Combining this data structure with a grouping trick (e.g. such as described by Chan [5]) yields the result in the lemma: S is partitioned into $\lceil n/r \rceil$ subsets $S_1, \dots, S_{\lceil n/r \rceil}$ of size at most r , and each S_i is processed into a data structure to answer dominance queries over S_i . The total preprocessing time is within $\mathcal{O}(\frac{n}{r}(r \log^{d-2} r)) = \mathcal{O}(n \log^{d-2} r)$. To answer whether a point q is dominated by any of the points in S , the data structures corresponding to each S_i are queried, for $i \in [1.. \lceil n/r \rceil]$: the final answer is “yes” only if q is dominated in any of the data structures. Thus, each query is answered in time within $\mathcal{O}(\frac{n}{r} \log^{d-2} r)$, for a total time within $\mathcal{O}(n \log^{d-2} r)$. \square

Note that, to detect whether a subset of \mathcal{S} falls within a box that is completely below the MAXIMA of \mathcal{S} (as in step 7 of DPC-Maxima), it is enough to query whether the corner with maximal coordinates (i.e. the “upper, right, . . .” corner) of the minimum axis-aligned rectangle of the subset is dominated by any point in \mathcal{S} . With this observation, and lemmas 1 and 2 we provide a first bound for the running time of DPC-Maxima:

Lemma 3. *Let \mathcal{S} be a set of points, and σ_j be the size of \mathcal{S} right after iteration j of the algorithm DPC-Maxima. The running time of DPC-Maxima is within $\mathcal{O}\left(\sum_{j=0}^{\log \log n} \sigma_j \times 2^{j(d-2)}\right)$.*

Proof. Let $r_j = 2^{2^j}$ (as defined in step 3 of DPC-Maxima). Obtaining the partition in step 4 using Lemma 1 costs time within $\mathcal{O}(|\mathcal{S}| \log r_j)$. Besides, the data structure described in Lemma 2 allows to perform the filtering steps in lines 5-7 collectively in total time within $\mathcal{O}(\sigma_j \log^{d-2} r_j)$ (using r_j as the parameter in the lemma). Applying the definition of r_j yields a total time within $\mathcal{O}(\sigma_j \times 2^{j(d-2)})$. Since at most $\log \log n$ iterations of the outer loop are performed, the result follows. \square

Finally, we show that DPC-Maxima runs in time sensitive to the structural entropy of the input set in the following theorem:

Theorem 1. *Let \mathcal{S} be a set of n points in \mathbb{R}^d , and let Π be any optimal respectful partition of \mathcal{S} into subsets $\mathcal{S}_1, \dots, \mathcal{S}_h$ of sizes n_1, \dots, n_h , respectively. The algorithm DPC-Maxima computes the MAXIMA of \mathcal{S} in time within $\mathcal{O}(n + \sum_{k=1}^h n_k \log^{d-2} \frac{n}{n_k})$.*

Proof. Let σ_j be the size of \mathcal{S} during iteration j . From Lemma 3 we have that the running time of DPC-Maxima is within $\mathcal{O}\left(\sum_{j=0}^{\log \log n} \sigma_j \times 2^{j(d-2)}\right)$. Now, consider a subset \mathcal{S}_k in Π , and let B_k be the minimum axis-aligned box enclosing \mathcal{S}_k . By Lemma 1.(ii) there are at most $\min\{n_k, \mathcal{O}(n/2^{2^j/d})\}$ points in \mathcal{S}_k that remain in \mathcal{S} after iteration j (remember that $r_j = 2^{2^j}$ is used as the parameter in the lemma). By summing over all the subsets in the partition, and plugging the previous bound for the point remaining in each subset after the j -th iteration of the outer loop, we have that the running time of the algorithm is within:

$$\begin{aligned}
& \sum_{j=0}^{\log \log n} \mathcal{O}(\sigma_j \times 2^{j(d-2)}) \\
& \subseteq \sum_{j=0}^{\log \log n} \sum_{k=1}^h \min\{n_k, \mathcal{O}(n/2^{2^j/d})\} \times 2^{j(d-2)} \\
& = \sum_{k=1}^h \sum_{j=0}^{\log \log n} \min\{n_k, \mathcal{O}(n/2^{2^j/d})\} \times 2^{j(d-2)} \quad (\text{reordering terms}) \\
& \in \sum_{k=1}^h \mathcal{O} \left(\sum_{j=0}^{\log(d \log \frac{n}{n_k})} n_k \times 2^{j(d-2)} + \sum_{j=1+\log(d \log \frac{n}{n_k})}^{\log \log n} \frac{n}{2^{\frac{1}{d}2^j}} \times 2^{j(d-2)} \right)
\end{aligned}$$

The left inner summation can be bounded by directly by $\mathcal{O}(n_k \log^{d-2} \frac{n}{n_k})$ ¹. To show that the right one is within $\mathcal{O}(n_k)$, start by bounding $2^{j(d-2)}$ by $2^{\frac{2^j-1}{d}}$, and take the infinite sum:

$$\sum_{j=1+\log(d \log \frac{n}{n_k})}^{\log \log n} \frac{n}{2^{\frac{2^j}{d}}} \times 2^{j(d-2)} \leq \sum_{j=1+\log(d \log \frac{n}{n_k})}^{\infty} \frac{n}{2^{\frac{2^j-1}{d}}}$$

This settles the case when $n_k > n/2$: when n is factored out, the remaining series converges, and the total summation is within $\mathcal{O}(n) \subseteq \mathcal{O}(n_k)$. For $n_k \leq n/2$, we substitute the variable of the sum by $i = j - \log(d \log \frac{n}{n_k}) - 1$ to obtain:

$$\begin{aligned}
& \sum_{j=1+\log(d \log \frac{n}{n_k})}^{\infty} \frac{n}{2^{\frac{2^j-1}{d}}} = \sum_{i=0}^{\infty} \frac{n}{2^{2^i \log n/n_k}} = \sum_{i=0}^{\infty} \frac{n}{n/n_k^{2^i}} \\
& = n_k \sum_{i=0}^{\infty} \frac{1}{(n/n_k)^{2^i-1}} \leq n_k \sum_{i=0}^{\infty} \frac{1}{(2)^{2^i-1}} \quad (\text{because } n/n_k \geq 2) \\
& \subseteq \mathcal{O}(n_k) \quad (\text{the serie converges})
\end{aligned}$$

Finally, replacing the bounds for the two inner summations yields the bound $\mathcal{O} \left(\sum_{k=1}^h n_k (1 + \log^{d-2} \frac{n}{n_k}) \right) \subseteq \mathcal{O} \left(n + \sum_{k=1}^h n_k \log^{d-2} \frac{n}{n_k} \right)$. \square

We prove next that, thanks to the concavity of the *polylog* function, the bound in Theorem 1 is never asymptotically worse than $\mathcal{O}(n \log^{d-2} h)$, and hence the running time of **DPC-Maxima** is never worse than the running time of Kirkpatrick and Seidel's algorithm [8]:

¹ Use the identity $\sum_{i=0}^m 2^{c \times i} = \frac{2^{c+m}-1}{2^c-1} \in \mathcal{O}(2^{cm}) \subseteq \mathcal{O}(\log^c m)$, for any constant c .

Lemma 4. *Let \mathcal{S} be a set of n points in \mathbb{R}^d , of MAXIMA of size h , and let Π be any optimal respectful partition of \mathcal{S} into subsets $\mathcal{S}_1, \dots, \mathcal{S}_h$ of sizes n_1, \dots, n_h , respectively. Then,*

$$\mathcal{O}\left(\sum_{k=1}^h n_k \log^{d-2} \frac{n}{n_k}\right) \subseteq \mathcal{O}\left(n \log^{d-2} h\right)$$

Proof. For any concave function φ , and $a = \sum_k a_k$, $b = \sum_k b_k$, Gibbs' inequality states that

$$\sum_i a_k \varphi\left(\frac{b_k}{a_k}\right) \leq a \varphi\left(\frac{b}{a}\right).$$

Since $f(x) = \log^{d-2} x$ is a concave function², choosing $a_k = \frac{n_k}{n}$, $b_k = 1$ for $k = [1..h]$ yields:

$$\sum_{k=1}^h \frac{n_k}{n} \log^{d-2} \left(\frac{n}{n_k}\right) \leq \log^{d-2} h.$$

Multiplying both sides of the inequality by n yields the result. \square

The upper bound in Theorem 1 properly generalizes Afshani et al.'s result [2] for dimensions 2 and 3 ($d \in [2, 3]$), but the lower bound proving is more tricky to prove: we discuss in the next section.

4 Discussion

The bound of $\mathcal{O}(n + \sum_{k=1}^h n_k \log^{d-2} \frac{n}{n_k})$ in Theorem 1 is within $\Theta(n \log^{d-2} h)$ for instances where all the subsets in an optimal respectful partition have equal size n/h (as illustrated in Figure 1.a). However, for instances with optimal unbalanced partitions (and hence with low structural entropy), the improvement in running time of DPC-Maxima over Kirkpatrick and Seidel's algorithm [8] can be significant. Consider, for example, an instance where all the points not in the MAXIMA are dominated by a same unique point (as the one illustrated in Figure 1.b), and let $h \in \mathcal{O}(n^{1-\varepsilon})$ be the size of the MAXIMA: Kirkpatrick and Seidel's algorithm [8] for this instance runs in time within $\mathcal{O}(n \log^{d-2} n)$, while DPC-Maxima runs in time within $\mathcal{O}(n)$, linear in the input size.

For the problem of MAXIMA in dimension $d \geq 3$, the best computational complexity lower bound known so far is $\Omega(n(\mathcal{H}(\mathcal{S}) + 1)) = \Omega(n + \sum_{k=1}^h n_k \log \frac{n}{n_k})$, obtained by extending the bound given in 2009 by Afshani et al. [2] for 2 dimensions. For dimension $d \geq 4$ there is a gap between this lower bound and the running time of the algorithm DPC-Maxima (which increases with d). The generalization of the order-oblivious instance-optimality result for the d -dimensional case for $d \geq 4$ is still open. In general, there is a lack of advanced lower bound

² $f(x)'' = d(d-1 - \log x)(\log^{d-2} x)/x^2 \leq 0$ for all $x \geq 2^{d-1}$

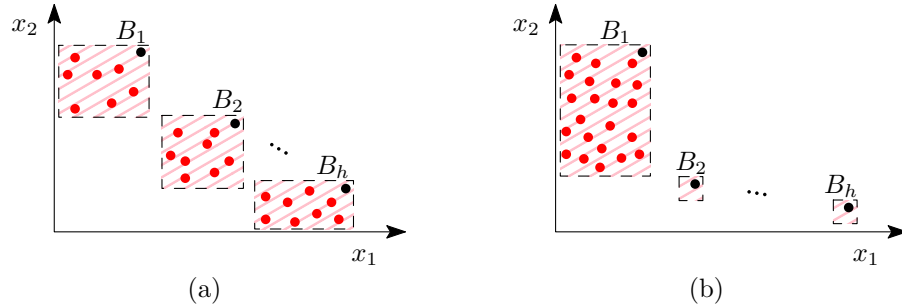


Figure 1: Two sets of n points and MAXIMA of size h with worst-case (a) and best-case (b) structural entropies, respectively. In both cases $\Pi = \{B_1, B_2, \dots, B_h\}$ is an optimal respectful partition, in (a) all the boxes contain n/h points, while in (b) B_1 contains $(n - h + 1)$ points, and B_2, \dots, B_h are singletons.

techniques for problems handling high dimensional data. In this sense, a theory of fine classes of problems (as partially done in the field of Parameterized Complexity [6]) could help to palliate this lack of techniques for lower bounds.

A closely related problem is DOMINANCE REPORTING, where for a set \mathcal{S} of n points in d -dimensional space, and a set of query points Q , one must report the points $q \in Q$ for which there is a point in \mathcal{S} that dominates q . For the offline version of the problem, where the size of Q is known to be significantly bigger than the size of \mathcal{S} , the fastest data structure known to date, in dimension $d \geq 3$, was presented in 2012 by Afshani et al. [1]. Karp et al. [7] considered an online version, where the size of Q is not known in advance, for which they describe a deferred data structure sensitive to the sizes of \mathcal{S} and Q . One natural question is whether the results by Karp et al. [7] for online DOMINANCE REPORTING can be improved via a deferred data structure sensitive to the structural entropy of the input set to query.

References

- [1] Peyman Afshani, Lars Arge, and Kasper Green Larsen. “Higher-dimensional orthogonal range reporting and rectangle stabbing in the pointer machine model.” In: *Symposium on Computational Geometry 2012, SoCG ’12, Chapel Hill, NC, USA, June 17-20, 2012*. Ed. by Tamal K. Dey and Sue Whitesides. ACM, 2012, pp. 323–332.
- [2] Peyman Afshani, Jérémy Barbay, and Timothy M. Chan. “Instance-Optimal Geometric Algorithms.” In: *50th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2009, October 25-27, 2009, Atlanta, Georgia, USA*. IEEE Computer Society, 2009, pp. 129–138.

- [3] J       Barbay, Ankur Gupta, Srinivasa Rao Satti, and Jonathan Sorenson. "Near-optimal online multiselection in internal and external memory." In: *J. Discrete Algorithms* 36 (2016), pp. 3–17.
- [4] Jon Louis Bentley. "Multidimensional Binary Search Trees Used for Associative Searching." In: *Commun. ACM* 18.9 (1975), pp. 509–517.
- [5] Timothy M. Chan. "Output-Sensitive Results on Convex Hulls, Extreme Points, and Related Problems." In: vol. 16. 4. 1996, pp. 369–387.
- [6] Rodney G. Downey and Michael R. Fellows. *Parameterized Complexity*. Monographs in Computer Science. Springer, 1999.
- [7] Richard M. Karp, Rajeev Motwani, and Prabhakar Raghavan. "Deferred Data Structuring." In: *SIAM J. Comput.* 17.5 (1988), pp. 883–902.
- [8] David G. Kirkpatrick and Raimund Seidel. "Output-size sensitive algorithms for finding maximal vectors." In: *Proceedings of the First Annual Symposium on Computational Geometry, Baltimore, Maryland, USA, June 5-7, 1985*. Ed. by Joseph O'Rourke. ACM, 1985, pp. 89–96.
- [9] H. T. Kung, Fabrizio Luccio, and Franco P. Preparata. "On Finding the Maxima of a Set of Vectors." In: *J. ACM* 22.4 (1975), pp. 469–476.